# Problem Set #4 Sample Solutions

Greg Plaxton

April 8, 2019

**1(a).** For any subset $U$ of $V$, let $f(U)$ denote $U \oplus X$. Thus $f$ is a bijection between the set of all feasible subsets of $V$ and the set of all good subsets of $V$. For any feasible subset $U$ of $V$, we have

$$\begin{aligned} v(U) &= w(X \cap U) - w(Y \cap U) \\ &= w(X) - w(X \cap f(U)) - w(Y \cap f(U)) \\ &= w(X) - w(f(U)). \end{aligned}$$

The claim of part (a) follows.

**1(b).** We construct a flow network $G' = (V', E')$ from $G$ as follows. We define the vertex set $V'$ as $V \cup \{s, t\}$. We define $E'$ as $E_1 \cup E_2 \cup E_3$ where $E_1 = \{(s, x) \mid x \in X\}$, $E_2 = E$, and $E_3 = \{(y, t) \mid y \in Y\}$. We define the capacity of each edge $(s, x)$ in $E_1$ as $w(x)$. We define the capacity of each edge in $E_2$ as $\infty$. We define the capacity of each edge $(y, t)$ in $E_3$ as $w(y)$. For any subset $U$ of $V$, let $g(U)$ denote the cut $(U + s, (V \setminus U) + t)$ of flow network $G'$.

Claim 1: For any subset $U$ of $V$ that is not good, the capacity of cut $g(f(U))$ is $\infty$. Let $U$ be a subset of $V$ that is not good. Thus $f(U)$ is infeasible, and hence $\Gamma(X \cap f(U))$ is not contained in $Y \cap f(U) = Y \cap U$. Thus there is an edge $(x, y)$ in $E_2$ such that $x$ belongs to $X \cap f(U)$ and $y$ does not belong to $Y \cap f(U)$. Hence $x$ belongs to the source side of cut $g(f(U))$ and $y$ belongs to the sink side of cut $g(f(U))$. Claim 1 follows since the capacity of edge $(x, y)$ is $\infty$.

Claim 2: For any good subset $U$ of $V$, the capacity of cut $g(f(U))$ is equal to $w(U)$. Let $U$ be a good subset of $V$. Thus $f(U)$ is feasible, and hence $\Gamma(X \cap f(U))$ is contained in $Y \cap f(U) = Y \cap U$. Thus no edges in $E_2$ contribute to the capacity of cut $g(f(U))$. Furthermore, the total capacity of the edges in $E_1$ (resp., $E_3$) that contribute to the capacity of cut $g(f(U))$ is $w(X \setminus f(U)) = w(X \cap U)$ (resp., $w(Y \cap f(U)) = w(Y \cap U)$). Thus the capacity of cut $g(f(U))$ is equal to $w(U)$, as required.

We can compute a minimum cut of $G'$ in polynomial time. Thus, we can determine a subset $U^*$ of $V$ such that $g(U^*)$ is a minimum cut of $G'$ in polynomial time. By Claims 1 and 2, $f(U^*) = U^* \oplus X$ is a minimum-weight good subset of $V$. The result of part (a) implies that $U^*$ is a maximum-value feasible subset of $V$.

**2.** Call a discharge operation *bad* if it results in a relabel, and *good* otherwise. Suppose the execution includes a contiguous sequence $S$ of $k$ good discharge operations, starting from a state $\sigma$. For $0 \leq i \leq k$, let $S_i$ denote the length-$i$ prefix of sequence $S$, let $V_i$ denote the set of vertices on which at least one of the discharge operations in $S_i$ is performed, and let $P(i)$ denote the predicate "$|V_i| = i$ and immediately after $S_i$ has been performed, no vertex in $V_i$ is overflowing". We now prove by induction on $i$ that $P(i)$ holds for all $i$ such that $0 \leq i \leq k$. The base case, $i = 0$, holds trivially. For the induction step, fix an integer $i$ such that $0 \leq i < k$, and assume that $P(i)$ holds. We need to prove that $P(i+1)$ holds. Let $\alpha$ denote the last discharge operation in $S_{i+1}$ and let $u$ denote the vertex on which $\alpha$ is performed. The IH implies that $u$ does not belong to $V_i$, and hence that $|V_{i+1}| = i+1$. Since no relabels are performed during execution of $S$, vertex heights do not change throughout execution of $S$. In what follows, when we refer to the height of a vertex, we mean its height throughout execution of $S$. Let $h$ denote the height of vertex $u$. Thus every vertex in $V_i$ has height at least $h$. It follows that no vertex in $V_i$ receives any flow as a result of the push operations associated with discharge operation $\alpha$. Since $u$ is not overflowing after operation $\alpha$, we deduce that no vertex in $V_{i+1}$ is overflowing after execution of $S_{i+1}$. This completes the proof by induction.

Since $P(k)$ holds, we conclude that $|S| \leq |V|$. The number of bad discharge operations is at most the number of relabel operations, and hence is $O(|V|^2)$. Since we can have at most $|V|$ good discharge operations in a row, the total number of discharge operations is $O(|V|^3)$. As in the case of the relabel-to-front algorithm, this implies that the number of nonsaturating pushes is $O(|V|^3)$. Since the algorithm under consideration is a push-relabel algorithm, the number of saturating pushes is $O(|V| \cdot |E|)$.

It remains to argue that the algorithm can be implemented to run in $O(|V|^3)$ time. The reasoning is similar to that used for relabel-to-front. Here we restrict our attention to the one new aspect, which is that in the present algorithm we need to repeatedly identify a maximum height overflowing vertex (to identify a suitable vertex to discharge). One natural way to address this issue is to maintain a priority queue containing all of the overflowing vertices organized by height. However this approach results in an overall $O(|V|^3 \log |V|)$ time bound. To eliminate the extra $\log |V|$ factor, we can instead use a simpler data structure. Specifically, we can maintain an ordered list of all of the vertices (whether overflowing or not) sorted in nonincreasing order of height, and a pointer into this list. Each time a relabel occurs, we can afford to spend $O(|V|)$ time reconstructing the list, and we reset the pointer to point to the first vertex in the list. When we need to find a maximum-height overflowing vertex, we simply advance the pointer until we find an overflowing vertex. This works because (by the inductive argument given earlier) we maintain the invariant that no vertex preceding the current pointer is overflowing. Furthermore, between successive relabel operations, the total time spent on identifying maximum-height overflowing vertices is $O(|V|)$. Since there are $O(|V|^2)$ relabel operations, the total time spent on identifying maximum-height overflowing vertices is $O(|V|^3)$.

**3.** We first show that all vertices with height exceeding $k$ are on the source side of a

minimum cut. For disjoint subsets of the vertices $A$ and $B$, let $c(A, B)$ denote the sum of the capacities of all edges going from a vertex in $A$ to a vertex in $B$, and let $g(A, B)$ denote the net flow from $A$ to $B$ induced by the current preflow $f$. That is, $g(A, B)$ is equal to $\sum_{u \in A} \sum_{v \in B} f(u, v) - f(v, u)$. Let $(S, T)$ denote an arbitrary cut. Let $X$ denote the set of all vertices in $T$ such that $h(v) > k$. (Note that $t$ does not belong to $X$ since $h(t) = 0 < k$.) In what follows, we complete the proof by showing that the capacity of cut $(S + X, T - X)$ is at most that of cut $(S, T)$. Now $c(S + X, T - X) - c(S, T) = c(X, T - X) - c(S, X)$, so it is sufficient to prove that $c(S, X) \geq c(X, T - X)$. We have $g(X, T - X) = c(X, T - X)$ since there are no edges from $X$ to $T - X$ with positive residual capacity. (Existence of such an edge implies that $h$ is not a height function.) Since $f$ is a preflow, $g(S + T - X, X)$ is nonnegative. Furthermore, $g(S + T - X, X) = g(S, X) + g(T - X, X) = g(S, X) - c(X, T - X) \leq c(S, X) - c(X, T - X)$. We conclude that $c(S, X) - c(X, T - X) \geq 0$, and hence $c(S, X) \geq c(X, T - X)$, as required.

It remains to argue that we still have a height function after applying the gap heuristic. Let $h$ be the height function before applying the gap heuristic, and let $h'$ be the function (to be shown to be a height function) that maps each vertex to its height after applying the gap heuristic. The residual capacities are unaffected by application of the gap heuristic. Let $(u, v)$ be an edge with positive residual capacity. We know that $h(u) \leq h(v) + 1$ since $h$ is a height function. Moreover, since neither $h(u)$ nor $h(v)$ is equal to $k$, and since $h(u) \leq h(v) + 1$, we find ourselves in one of the following two cases.

Case 1: Both $h(u)$ and $h(v)$ are less than $k$. In this case, $h'(u) = h(u)$ and $h'(v) = h(v)$, so $h'(u) \leq h'(v) + 1$, as required.

Case 2: $h(v)$ is greater than $k$. In this case, $h'(u) \leq \max(h(u), \Delta)$ and $h'(v) = \max(h(v), \Delta)$ where $\Delta = |V| + 1$. Now $h(u) \leq h(v) + 1 \leq h'(v) + 1$ and $\Delta \leq h'(v)$, so $h'(u) \leq h'(v) + 1$, as required.

**4(a).** The augmenting path $P$ visits some sequence of vertices $\langle x_0, \ldots, x_{2k+1} \rangle$, $k \geq 0$, where the even-indexed vertices belong to $L$ and the odd-indexed vertices belong to $R$. Note that $M$ consists of the $k$ edges $(x_{2i+1}, x_{2i+2})$, $0 \leq i < k$, and each of the vertices except the endpoints of $P$, namely $x_0$ and $x_{2k+1}$, is matched (i.e., is an endpoint of exactly one matched edge). The symmetric difference $M \oplus P$ consists of the $k + 1$ edges $(x_{2i}, x_{2i+1})$, $0 \leq i \leq k$. Note that every vertex is an endpoint of exactly one edge in $M \oplus P$, so it is a matching. The generalization to $k$ vertex-disjoint augmenting paths $P_1, \ldots, P_k$ is equally straightforward. (Vertex-disjointness of the paths guarantees that no vertex is an endpoint of more than one edge in $M' = M \oplus (P_1 \cup \cdots \cup P_k)$, so that $M'$ is a matching. Furthermore, $M'$ contains one more edge than $M$ in each of the paths $P_i$, so $|M'| = |M| + k$.)

**4(b).** Since $M$ and $M^*$ are matchings, every vertex has degree at most one in each of the graphs $(V, M)$ and $(V, M^*)$. Hence every vertex has degree at most two in the multigraph $(V, M \cup M^*)$. (A multigraph is a graph in which parallel edges are allowed.) Since the graph $G' = (V, M \oplus M^*)$ is a subgraph of the latter graph, every vertex has degree at most two in $G'$. It follows easily that $G'$ is a disjoint union of simple paths and cycles. (To see this, find a vertex $v$ with nonzero degree and greedily grow a maximal path from $v$. Note that

the growth process can only terminate when we end up with a maximal path or a cycle. In particular, we can never return to a nonendpoint of the path we are growing since that would imply the existence of a vertex with degree at least 3. Once we have obtained a maximal path or a cycle, we can remove the edges of that path or cycle, and repeat the process on the remaining graph.) Because $M$ is a matching, no two consecutive edges on any of these paths or cycles belong to $M$. Similarly, no two consecutive edges belong to $M^*$. But every edge on these paths and cycles belongs to either $M$ or $M^*$, so the edges alternate between $M$ and $M^*$. It follows that every cycle is of even length, and contributes an equal number of edges to $M$ and $M^*$. Note that the number of edges on each path $P$ that belong to $M$ is within one of the number that belong to $M^*$. Thus, if $|M^*| = |M| + k$, $k \geq 0$, at least $k$ of the paths have one more edge in $M^*$ than in $M$. Furthermore, each of these vertex-disjoint paths is an augmenting path with respect to $M$, since it begins and ends with an edge in $M^*$.

**4(c).** If the length of $P$ is less than $\ell$, then $P_1, \ldots, P_k$ are not shortest augmenting paths, a contradiction. If the length of $P$ is $\ell$, then $\{P_1, \ldots, P_k\}$ is not maximal, a contradiction. Hence, the length of $P$ is more than $\ell$.

**4(d).** That $A$ is equal to $(P_1 \cup \cdots \cup P_k) \oplus P$ is immediate from the definitions of $A$ and $M'$, and basic properties of the $\oplus$ operator (e.g., associativity, $M \oplus M$ is the empty set). Note that $A$ is also equal to $M \oplus M^*$ where $M^* = M' \oplus P$. By part (a), $|M'| = |M| + k$ and $|M^*| = |M'| + 1$; thus, $|M^*| = |M| + k + 1$. By part (b), $A$ contains at least $k + 1$ vertex-disjoint augmenting paths with respect to $M$. By the definition of $\ell$, each of these paths has length at least $\ell$. Thus $|A| \geq (k+1)\ell$. Since $|P_1 \cup \cdots \cup P_k| = k\ell$ and $A = (P_1 \cup \cdots \cup P_k) \oplus P$, at least $\ell$ edges of $P$ do not belong to $P_1 \cup \cdots \cup P_k$. Furthermore, since $P$ is not vertex disjoint from $P_1, \ldots, P_k$, there is a vertex $u$ in $P$ that belongs to some $P_i$, $1 \leq i \leq k$. Note that $u$ is matched in $M'$, i.e., there is an edge $(u, v)$ in $M'$. Furthermore, the edge $(u, v)$ belongs to both $P$ and $P_i$. Hence $P$ contains at least one edge that belongs to $P_1 \cup \cdots \cup P_k$. It follows that $P$ has more than $\ell$ edges.

**4(e).** Let $M^*$ be a maximum matching. By part (b), $M \oplus M^*$ contains at least $|M^*| - |M|$ vertex-disjoint augmenting paths with respect to $M$. Furthermore, by the assumption that a shortest augmenting path for $M$ has length $\ell$, each of these $|M^*| - |M|$ vertex-disjoint augmenting paths has length at least $\ell$. Now a length-$\ell$ path contains $\ell + 1$ vertices, so the total number of vertices in these paths is at least $(|M^*| - |M|)(\ell + 1)$. Since this number of vertices is at most $|V|$, we find that $|M^*| - |M| \leq \frac{|V|}{\ell+1}$. Thus $|M^*| \leq |M| + \frac{|V|}{\ell+1} \leq |M| + \frac{|V|}{\ell}$.

**4(f).** Using the results of parts (c) and (d), we can easily prove by induction that after $k$ iterations of the loop, the length of a shortest augmenting path is at least $k + 1$. Therefore, after $\sqrt{|V|} - 1$ iterations, the shortest augmenting path is of length at least $\sqrt{|V|}$, and by part (e), the number of augmentations yet to be performed is at most $\sqrt{|V|}$. Since each iteration of the loop except the last iteration performs at least one augmentation, at most

$\sqrt{|V|}+1$ iterations remain to be performed. Thus the total number of iterations is at most $2\sqrt{|V|}$.

**4(g).** The desired conclusion is immediate from part (f) once we establish that each iteration can be implemented in $O(|E|)$ time. Our implementation uses a two-phase approach. In the first phase, we determine the length $\ell$ of a shortest augmenting path and label every vertex $u$ that is reachable via an alternating (edge not in the current matching, edge in the current matching) path of length at most $\ell$ from some unmatched vertex in $L$ with the length of the shortest such path to $u$. (Exception: If the preceding procedure would assign a label of $\ell$ to some vertex $v$, then we do not actually assign the label to $v$ unless it is unmatched.) This labeling is accomplished in stages numbered from 0 to $\ell$; in the $i$th stage, the vertices that should receive a label of $i$ are identified and labeled. (We remark that vertices in $L$ can only receive even-numbered labels, and vertices in $R$ can only receive odd-numbered labels.) In stage 0, every unmatched vertex in $L$ is assigned a label of 0. In stage $2k+1$, $k \geq 0$, the label $2k+1$ is assigned to every unlabeled vertex $v$ in $R$ for which there exists a vertex $u$ in $L$ with label $2k$ such that $(u, v)$ belongs to $E \setminus M$. Furthermore, if in stage $2k+1$ we have assigned a label to an unmatched vertex, then we set $\ell$ to $2k+1$, remove the label from every matched vertex that received label $\ell$, and terminate the first phase. In stage $2k+2$, $k \geq 0$, the label $2k+2$ is assigned to every unlabeled vertex $u$ in $L$ for which there exists a vertex $v$ in $R$ with label $2k+1$ such that $(u, v)$ belongs to $M$. An important note is that if we fail to label any vertex at some stage then we report that there are no augmenting paths with respect to $M$, so Hopcroft-Karp terminates and reports $M$ as a maximum matching. The graph search performed in the first phase, which is sometimes referred to as "alternating breadth-first search (BFS)", may be implemented in a manner similar to ordinary BFS. This implementation runs in $O(|E| + |X|)$ time. Assuming that we use a simple preprocessing phase at the start of the overall matching algorithm to remove any degree-zero vertices, this time bound is $O(|E|)$.

In the second phase, we construct a maximal set $S$ of vertex-disjoint length-$\ell$ augmenting paths. Our plan is to initialize $S$ to the empty set, and then repeatedly identify an augmenting path $P$ that is vertex-disjoint from all paths in $S$, and add $P$ to $S$. Let $X$ denote the set of (unmatched) vertices with label 0, and let $Y$ denote the set of unmatched vertices with label $\ell$. Note that every augmenting path connects some vertex in $X$ with some vertex in $Y$. Furthermore, given the labeling that we performed in the first phase, we can start at any vertex in $Y$ and greedily grow a path backwards to some vertex in $X$ (visiting a vertex with a label that is smaller by one at each step). Hence it is easy to find a first augmenting path. The difficulty that we need to overcome is that after we have identified one or more vertex-disjoint augmenting paths, it is no longer clear how to efficiently determine another augmenting path that is vertex-disjoint from those we have already found: If we simply try to start at any vertex in $Y$ and greedily grow a path backwards to some vertex in $X$, we might run into a vertex belonging to some path that is already in $S$.

Just as we have used an alternating variant of BFS to implement the first phase, there is an elegant way to implement the second phase using an alternating variant of depth-first

search (DFS). We proceed in $|Y|$ stages, and in each stage we run a modified DFS from a distinct vertex in $Y$. Before the first stage, we ensure that all vertices are "unmarked". Our DFS routine "marks" every vertex that it visits. When searching from a vertex with label $i$ such that $0 < i \leq \ell$, we only traverse outgoing edges that lead to unmarked vertices with label $i - 1$. If the DFS performed in a stage ever visits a vertex in $X$ (i.e., a vertex with label 0), then we have successfully found another augmenting path and we terminate the stage. For any $i$ such that $0 \leq i \leq |Y|$, let $\mathcal{P}_i$ denote the set of augmenting paths discovered in the first $i$ stages. It is easy to prove by induction on $i$ that if a vertex $v$ is marked during the first $i$ stages, then there is no augmenting path $P$ that contains $v$ and is vertex-disjoint from all of the augmenting paths in $\mathcal{P}_i$. It follows that if we search from a vertex $y$ in $Y$ in stage $i + 1$, where $0 \leq i < |Y|$, and if $y$ lies on an augmenting path that is vertex-disjoint from all of the augmenting paths in $\mathcal{P}_i$, then the search from $y$ is guaranteed to find such an augmenting path. The running time of the second phase is $O(|E| + |V|)$ because we do not visit any vertex or edge more than once. This bound simplifies to $O(|E|)$ because each vertex in $V \setminus X$ that we visit has nonzero degree, and we visit at most $|Y|$ vertices in $X$.